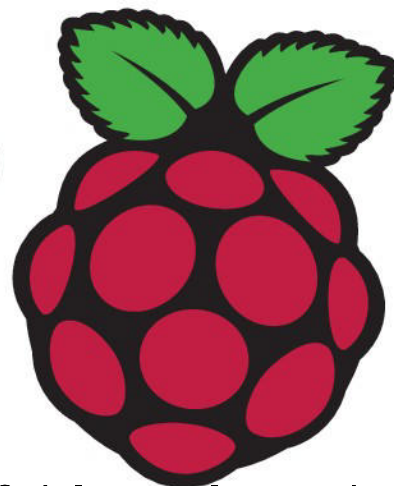
VISITA WWW.RASPBERRYITALY.COM

The MagPi



Numero 79 | Marzo 2019 | magpi.cc raspberryitaly.com

La rivista ufficiale Raspberry Pi tradotta in italiano da RaspberryItaly

SVILUPPA UNA APP ANDROID

Crea e installa la tua app con Raspberry Pi

SPECIALE MAKER!

#MONTHOFMAKING

11 progetti da creare e condividere

PROGRAMMA UN GIOCO ISOMETRICO

La seconda parte del tutorial sulla creazione del gioco AmazeBalls



The MagPi magazine is published by Raspberry Pi (Trading) Ltd., Mount Pleasant House, Cambridge, CB3 0RN. ISSN: 2051-9982.

SPECIALE MAKER!

#MONTHOFMAKING

Costruisci il tuo primo progetto o mostra i tuoi progressi in questo evento online della community!

La community di Raspberry Pi è enorme. È una delle cose più sorprendenti di tutte quelle che costituiscono il Raspberry Pi - una collezione mondiale di persone entusiaste di mostrare quello che si può fare con il Raspberry Pi, e anche di aiutare gli altri con i loro progetti ai Raspberry Jam, Code Club, CoderDojo e altri incontri!

Vogliamo celebrare la community durante il mese di Marzo con il nostro evento #MonthOfMaking: per tutto il mese, lavora su un progetto e condividi i tuoi lavori in corso sui social media. Questo è tutto! Sia che tu sia nuovo nel making o che tu abbia un Raspberry Pi che ti prepara la colazione, vogliamo vedere cosa puoi fare e inondare internet con usi creativi per l'elettronica fisica.

Hai bisogno di alcune idee su come iniziare? Continua a leggere per avere qualche ispirazione...

Strumenti del mestiere

Per fare qualcosa di interessante, potresti aver bisogno di alcuni strumenti

Elettronica

I circuiti possono collegare il Raspberry Pi e i microcontrollori a luci, pulsanti, altoparlanti o anche a un robot completo. Ecco alcuni degli strumenti del mestiere.

- Fili
- Cavi a cavallotto
- Breadboard
- LED
- Pulsanti
- Motori
- Stagnatore

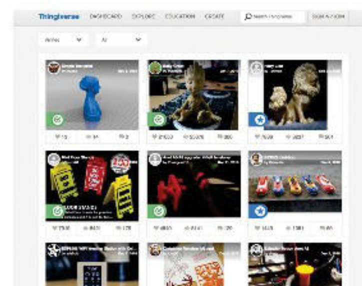
Legno e metallo

I metodi di costruzione vecchia scuola a volte possono richiedere un po' di attrezzatura seria. Se sei nuovo, chiedi a un adulto esperto alcuni suggerimenti.

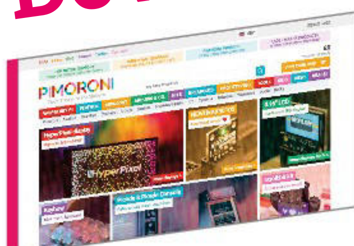
- Sega
- Saldatrice
- Morsa
- Trapano
- Colla
- Levigatrice
- Seghetto
- Strumenti di misura

Stampe 3D e plastiche

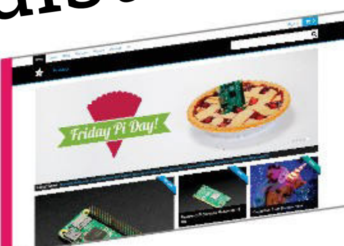
Don't try this at home... A meno che tu non abbia una costosa stampante 3D o taglierina laser. Se non le hai, puoi trovare progetti per stampanti 3D su siti Web come Thingiverse (**thingiverse.com**) e stamparli su 3D Hub (**3dhubs.com**).



Dove acquistare



Pimoroni
shop.pimoroni.com
Un fornitore del Regno Unito di Raspberry Pi e tecnologia maker



Adafruit
adafruit.com
Un leader globale in fantastici componenti maker e hardware



eBay
ebay.co.uk
Fornitore di componenti economici per praticamente ogni cosa tu voglio fare



Idee per aspiranti maker

Sei nuovo nel making? Ecco alcune idee che ti aiuteranno

Tutti devono iniziare da qualche parte. Tutti i maker più sorprendenti del mondo a un certo punto hanno dovuto capire come visualizzare 'Ciao mondo', come controllare un LED con il codice e come costruire.

Non aver paura di condividere le tue prime incursioni nel mondo del making. Il resto di noi è qui per aiutare e apprezzare il tuo lavoro! Per darti quel primo impulso, ecco alcuni progetti per principianti che potresti trovare utili.



01 Indossabili da principiante Sushi Cards

Questo è un divertente progetto indossabile che usa semplicemente circuiti e parti pre-progettate per creare alcuni articoli di abbigliamento dall'aspetto divertente. Perfetto per mostrare le tue nuove abilità di maker!
magpi.cc/ZjUjiS



02 Whoopi Cushion

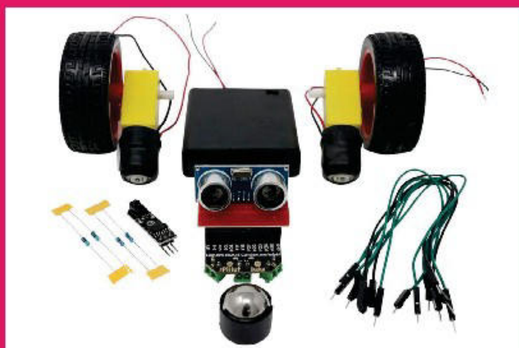
Che senso ha fare making se non ti diverti a farlo? Amiamo particolarmente questo progetto perché è molto economico da realizzare e include alcune vere abilità da fai da te per creare un grande pulsante per il dispositivo.
magpi.cc/2AgN6W



03 Timer conto alla rovescia

Questo semplice progetto ti mostra come usare un Sense HAT – un accessorio per il Pi con sensori ambientali e un semplice display - come un timer per il conto alla rovescia. Probabilmente puoi anche trasformarlo in un cronometro.
magpi.cc/KnaWBY

Kit divertenti



CamJam EduKit 3 | 21€

Questo economico kit ti consente di utilizzare il tuo Raspberry Pi per creare un robot che può essere completamente automatizzato - la scatola stessa può essere usata come un telaio! È anche supportato in GPIO Zero, la pratica libreria di programmazione Python per il physical computing che viene fornita con Raspbian su Raspberry Pi.
magpi.cc/RhpjZh

Mood Light | 37€



A differenza di alcuni altri kit, questo viene fornito con un Pi Zero W e praticamente tutto quel che serve a costruirlo. Insegna la programmazione di base, e ha anche dei buoni componenti che puoi utilizzare per progetti futuri.
magpi.cc/xqySvs



Console Picade | 70€

Un kit leggermente più costoso, ma molto divertente: crea la tua console di gioco retrò nel formato di un classico joystick arcade con pulsanti! È una grande realizzazione, e alcuni componenti sono davvero utili per progetti futuri.
magpi.cc/BSeTDD

Risorse educative da Raspberry Pi

Parte della missione della Fondazione Raspberry Pi è quella di rendere accessibile gratuitamente l'istruzione informatica a tutti e il sito Raspberry Pi projects ha tutorial su programmazione, making e altro per te, per imparare. Puoi trovarli su **rpf.io/projects**, dove troverai ancora più idee maker mentre esplori tutorial su "Physical computing with Python" (**magpi.cc/2mjGMJZ**) o "Getting started with picamera" (**magpi.cc/QrMnng**).

Maker di ispirazione

Ecco un po' di gente da seguire su YouTube!



Estefannie Explains It All

Ingegnere software e maker di progetti divertenti
magpi.cc/vDhJBq



Jabrils

Maker specializzato in AI
magpi.cc/foAxEd



Tinkernut

Making, riparazioni e hacking
magpi.cc/pQNeif



Junie Genius

Scienza, making, e tutto quel che c'è in mezzo
magpi.cc/oSkvRG



Blitz City DIY

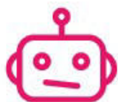
Un maker che si diletta un po' più nel lato artistico
magpi.cc/FDpqSi

Idee per maker hobbisti

Se questo non è il tuo primo rodeo maker, ecco alcuni progetti divertenti da provare

Scegli un tema!

Non sai dove iniziare? Scegliere un tema per la tua realizzazione/i, ti aiuterà!



Robot



IoT



Home media



Automazione domestica



Ufficio domestico



Foto e video

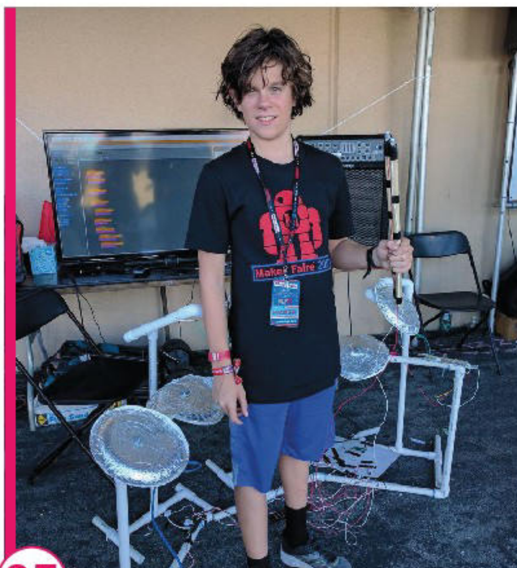
Ogni giorno, vediamo ancora un altro incredibile progettare o rifacciamo ciò che qualcuno ha già realizzato. Sembra sia quasi impossibile rimanere senza idee - ma a volte abbiamo bisogno di un poco di ispirazione. Ecco alcuni progetti per accendere le tue cellule cerebrali.



04 Tende controllate da Amazon Echo

IA e controllo vocale sono ampiamente accessibili su Raspberry Pi, grazie ad Alexa di Amazon, Google AIY e persino Cortana di Microsoft. Questo progetto si sposa con alcune automazioni per aprire e chiudere le tende a casa tua. Puoi adattarlo per controllare altri dispositivi automatici, come le luci o la porta del garage.

magpi.cc/AEWhkN



05 Raspberry Pi Drum Kit

Questo progetto è il massimo del fai da te ed è anche molto economico da fare! Usa un sacco di tubi in PVC, piatti di carta, fili e molte altre minuterie che puoi trovare in un negozio di bricolage. Oh, e richiede anche due dadi ciechi. È programmato in Scratch, rendendolo molto accessibile ai giovani maker.

magpi.cc/JzdHuD



06 DiddyBorg v2

Che ne dici di costruire un robot più grande? DiddyBorg v2 è uno dei nostri kit robot preferiti, e puoi personalizzarlo molto pesantemente per fare qualche cosa straordinaria in automatico. La cosa migliore è che tutte le parti sono di alta qualità e facilmente trasferibili anche su robot più grandi e migliori!

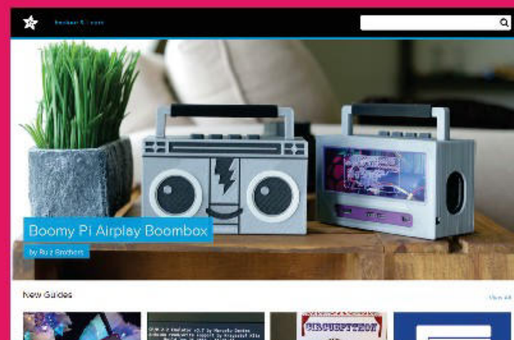
magpi.cc/SHBgNc

07 Magic Mirror

Questo classico progetto è quasi un rito di passaggio per i possessori di Raspberry Pi. È molto più semplice di quanto sembri, anche grazie all'ottimo software di Michael Teeuw. Vi abbiamo fatto un grande speciale sul numero 54 di *The MagPi* (**bit.ly/2rT7zgu**), e ora puoi trovare un sacco di funzioni diverse da aggiungere a questo tipo di progetto.

magicmirror.builders

Risorse per i maker



Adafruit Learning System

Su Adafruit, oltre a acquistare ottimo hardware, puoi trovare grandi progetti con fantastiche e approfondite istruzioni su come costruirli. C'è sempre qualcosa di bello e di nuovo sul sito web, che copre una vasta gamma di diversi tipi di progetti!

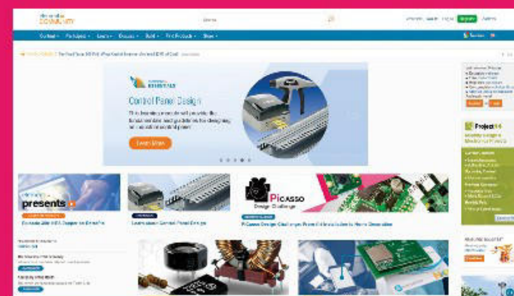
learn.adafruit.com

Rivista HackSpace



La nostra pubblicazione sorella, La rivista HackSpace è una risorsa incredibile per realizzare cose incredibili, e non solo con il Raspberry Pi. Ovviamente, abbiamo anche grandi progetti e tutorial, ma tu hai già letto ogni numero di *The MagPi*. Giusto?

hsmag.cc



Community element14

Non solo un rivenditore Raspberry Pi, ma anche una straordinaria comunità di maker che partecipano a sfide e mostrano ciò che hanno fatto. Sicuramente vale la pena dare un'occhiata a ciò che è stato realizzato di recente, e a qual è il tema del mese!

magpi.cc/GRmbXe

Idee per maker esperti

Pronto una vera sfida? Prepara il tuo saldatore ad arco...

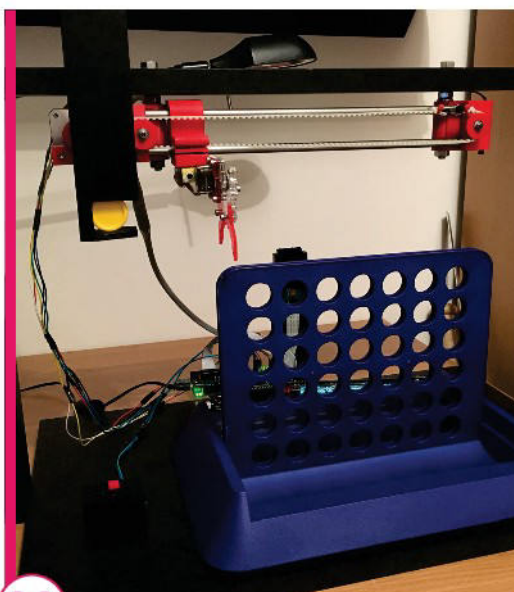
A tutti piace una realizzazione mostruosa. Qualcosa di impressionante. Se sei il tipo di maker che costruisce furgoni-drago sputafuoco nel sonno, allora ecco alcune sfide fantastiche di cui potresti essere all'altezza.



08 Arcade cabinet

Certo, puoi usare un kit per creare una piccola scatola da appoggiare al bancone, ma per costruire un cabinet arcade completo da zero, con luci funzionanti. decorazioni e tutto, ci vuole un po' di tempo e di abilità. Ci piace particolarmente questa versione fatta da I Like To Make Stuff.

magpi.cc/sniEhi



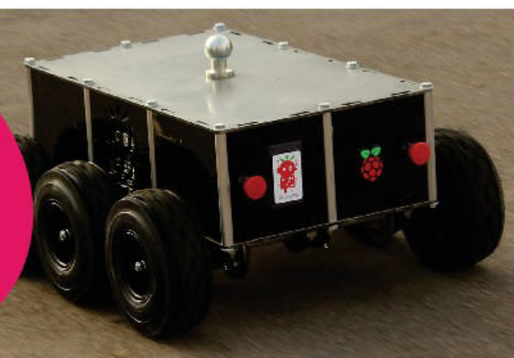
09 4-Bot

I robot non sono solo tutti i piccoli veicoli guidabili che puoi comprare in kit - ci sono anche bracci robotici, robot bipedi e altri tipi di robot camminatori. Ci piace molto questo 4-Bot: è un robot con cui puoi giocare a Forza 4. Usa la computer vision per vedere come hai giocato e calcola la sua prossima mossa. Abbiamo anche visto mani robotiche che giocano a scacchi. Finché tu puoi programmarlo, puoi realizzarlo.

magpi.cc/1XrC3zU

Più GRANDE!

Il DoodleBorg è un robo-bestia a sei ruote fatto da PiBorg. È abbastanza potente da rimorchiare una roulotte!
magpi.cc/ieGAjS



Diventa competitivo

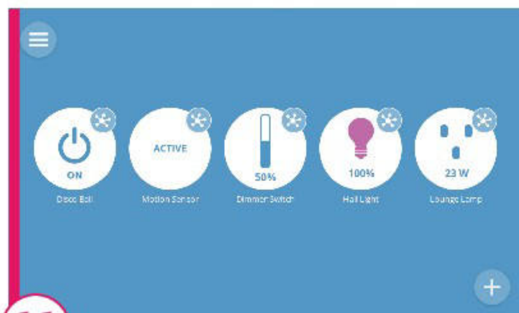


Più GRANDE!

OpenROV crea incredibili droni sottomarini alimentati dal Pi chiamati Trident.
openrov.com

10 Drone sottomarino/ROV

Ogni volta che mischi acqua ed elettronica, avrai sempre dei momenti difficili. Il vero trucco con un ROV sommergibile è quello di mantenere l'elettronica all'asciutto, pur essendo ancora in grado di controllarlo da remoto. Oh, e abbiamo menzionato i problemi di bilanciamento nel tenere qualcosa sott'acqua? È una procedura abbastanza precisa, che lo rende perfetto per una grande sfida.
magpi.cc/NLAiTh



11 Smart home

Cosa ne dici di controllare la tua casa attraverso un Raspberry Pi? Abbiamo visto alcune persone tentare, con risultati variabili di successo. Il Project Things di Mozilla è un buon modo per usare la tecnologia per automatizzare completamente la tua casa - dagli un'occhiata!
magpi.cc/avfrFj

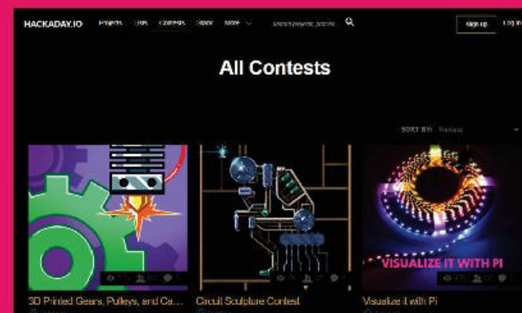


Pi Wars

La competizione annuale per i robot Raspberry Pi dei maker coinvolge molte competizioni non violente, in modo che il tuo robot possa tornare a casa con tutta la sua verniciatura originale e/o ruote. Forse è anche un po' troppo tardi per iscriversi, ma il prossimo anno sarà qui prima che tu te ne renda conto.
piwars.org

Makevember

#MonthOfMaking non è il primo hastang incentrato sui maker: se vuoi davvero una sfida, #makevember in Novembre assegna ai maker dei compiti per fare qualcosa ogni giorno! Sebbene probabilmente non realizzerai nulla di simile a quello che potresti fare in oltre un mese, rimane divertente per sperimentare ed essere creativo.



Hackaday

Oltre ad essere un ottimo repository per progetti interessanti e realizzazioni, ci sono gare regolari in cui chiunque può entrare. C'è persino un premio annuale per il miglior maker! Se stai realizzando qualcosa di veramente spettacolare, non perdere l'occasione di metterlo qui.
hackaday.io/contests

Parte 02

Programmare un gioco isometrico: AmazeBalls



MAKER

Mark Vanstone

Autore di software educativo degli anni '90, autore della serie ArcVenture, scomparso nella landa desolata del software aziendale. Salvato dal Raspberry Pi!

magpi.cc/YiZnxL

@mindexplorers

Pygame Zero in 3D. Ingrandiamo la mappa nella seconda parte di questo tutorial in tre puntate

In questa seconda parte del tutorial sull'uso di Pygame Zero per creare un gioco 3D isometrico, partiremo da dove ci eravamo interrotti la volta scorsa e vedremo come rendere la nostra area 3D più grande e più facilmente modificabile. Questo significherà usare un editor di mappe chiamato Tiled, gratuito da scaricare ed usare, per creare le nostre mappe 3D. Impareremo a creare una semplice mappa ed esportarla da Tiled in un formato dati chiamato JSON. Verrà poi importata nel gioco e codificherà la funzione di disegno per scorrere intorno all'area di movimento del giocatore. C'è tanto da fare, quindi iniziamo.

Cosa Serve

- Raspbian Jessie o più recente
- Tiled (editor di mappe gratuito) **mapeditor.org**
- Un programma di manipolazione immagini, come GIMP, o le immagini su **magpi.cc/fPBrhM**
- L' ultima versione di Pygame Zero (1.2)

01 Strumenti di lavoro

Nella parte precedente del tutorial, abbiamo realizzato la nostra mappa scrivendo una lista bidimensionale di "zero" e "uno" per rappresentare un blocco di pavimento o un blocco di muro. Il giocatore si poteva muovere in qualunque blocco "zero" dei dati. Ora abbiamo una app editor di mappe che farà il lavoro per noi, invece di inserirle i dati a mano. Innanzitutto dobbiamo installare Tiled. Puoi trovare l'home page di Tiled su *6 &9*) o8;h8;, ,* , dove c'è la possibilità di supportare gli sviluppatori, se ti piace quello che stanno creando.

02 Ottenere Tiled

Tiled può essere usato su sistemi diversi, inclusi PC e computer Mac. Inoltre, cosa molto importante per noi, lavora molto bene con Raspbian e Raspberry Pi. È anche estremamente facile da installare. Tutto quel che devi fare è aprire una finestra del Terminale, assicurarti di essere on-line e aggiornare il sistema digitando `sudo apt-get update`. Potrebbe essere necessario digitare anche `sudo apt-get upgrade`, a seconda di quando tempo è passato dall' ultima volta che hai aggiornato il Pi. Una volta terminato, basta digitare `sudo apt-get install tiled`, premere **INVIO**, e si dovrebbe avere Tiled installato. Quando terminato, apparirà una nuova icona nel sotto menù Grafica.

03 Cartografia

Precedentemente la mappa era di 12 blocchi per 12, e ci stava tutta nello schermo. Con l'editor si può costruire una mappa molto più grande. Potrebbe essere enorme, ma per il momento ci limitiamo ad una griglia di 30 per 30 blocchi. Puoi scaricare la mappa già pronta da *6 &; 9d((a+TF;. Q .* Se carichi la mappa vedrai un labirinto rosso e blu, come nella prima parte, ma molto più grande. Questa volta, però, abbiamo aggiunto un blocco per indicare il punto di uscita dal labirinto. Dovresti essere in grado di scorrere per vedere l'intera mappa.

04 Esportare i dati

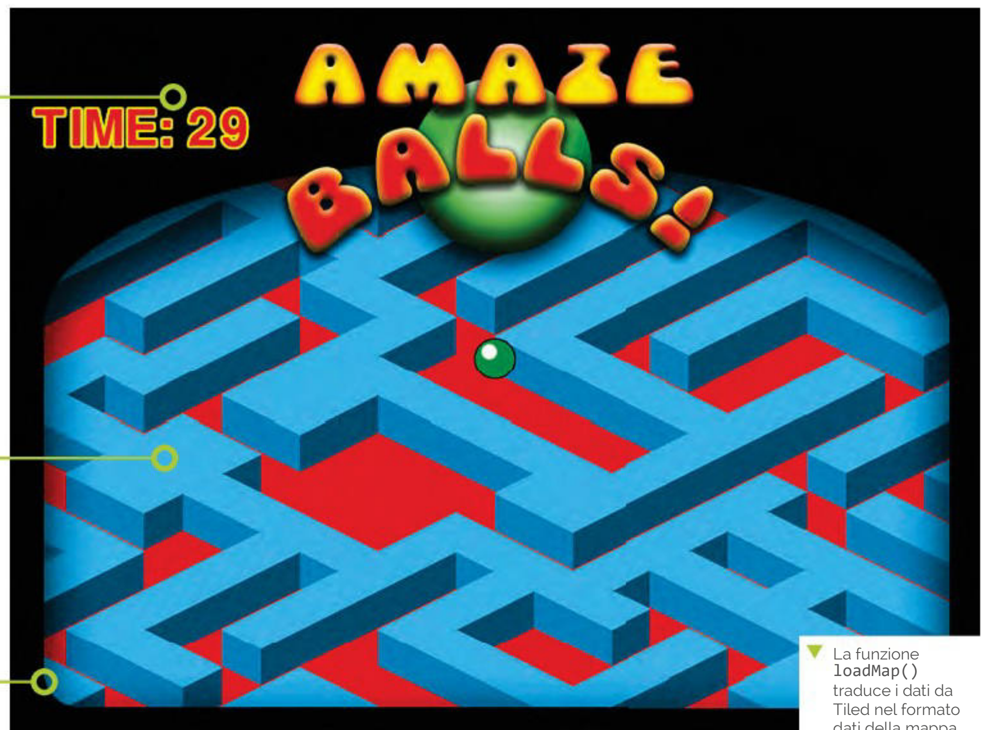
Gioca con l'editor di mappe; c'è una buona documentazione sul sito web. Quando hai familiarizzato sul suo funzionamento, potrai pensare a come ottenere la mappa per il nostro gioco. Per esportare i dati, vai su Export nel menù File e quando si apre la finestra di dialogo, che chiede 'export as...', cerca dove salvare (forse in una sottodirectory **maps**) la mappa (forse come **map1**), ma nel menù a discesa con l'etichetta 'Save as type', seleziona 'Json map files (*.json)'. Questo tipo di file (pronunciato 'Jay-son', abbreviazione per JavaScript Object Notation) può essere letto con un editor di testo.



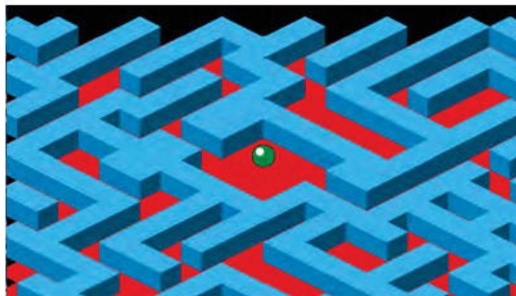
Il giocatore deve completare il labirinto il più velocemente possibile

E' stato usato un editor esterno per creare il labirinto

Il labirinto è più grande della finestra di gioco e si sposta intorno con il giocatore



La funzione `loadMap()` traduce i dati da Tiled nel formato dati della mappa



Quando abbiamo riscritto la nostra funzione `drawMap()`, vedremo alcuni bordi frastagliati attorno alle estremità dell'area di disegno

05 JSON e gli Argonauti

Se apriamo un file JSON, vedremo una serie di parentesi graffe e quadre con parole e numeri sparsi dappertutto, ma prima di proclamare 'Questo per me è arabo!', diamo un'occhiata ad alcuni elementi in modo da capire la struttura dei dati. Se hai familiarità con il linguaggio JavaScript, riconoscerai che le parentesi graffe { e } sono usate per delimitare il blocchi di codice o dati, e le parentesi quadre [e] sono usate per le liste di dati. Guarda l'elemento chiamato 'layers' e vedrai i dati che descrivono la nostra mappa.

06 Caricare i dati

Per questo gioco non abbiamo bisogno di tutti i dati presenti nel file JSON, ma possiamo caricarlo tutto e usare solo i bit di cui abbiamo

figure1.py

```
import json
import os

def loadmap(mp):
    with open(mp) as json_data:
        d = json.load(json_data)
    mapdata = {"width":d["width"], "height":d["height"]}
    rawdata = d["layers"][0]["data"]
    mapdata["data"] = []
    for x in range(0, mapdata["width"]):
        st = x*mapdata["width"]
        mapdata["data"].append(rawdata[st:st+mapdata["height"]])

    tileset = "maps/" + d["tilesets"][0]["source"].replace(".tsx",".json")
    with open(tileset) as json_data:
        t = json.load(json_data)

    mapdata["tiles"] = t["tiles"]
    for tile in range(0,len(mapdata["tiles"])):
        path = mapdata["tiles"][tile]["image"]
        mapdata["tiles"][tile]["image"] = os.path.basename(path)
        mapdata["tiles"][tile]["id"] = mapdata["tiles"][tile]["id"]+1
    return mapdata
```

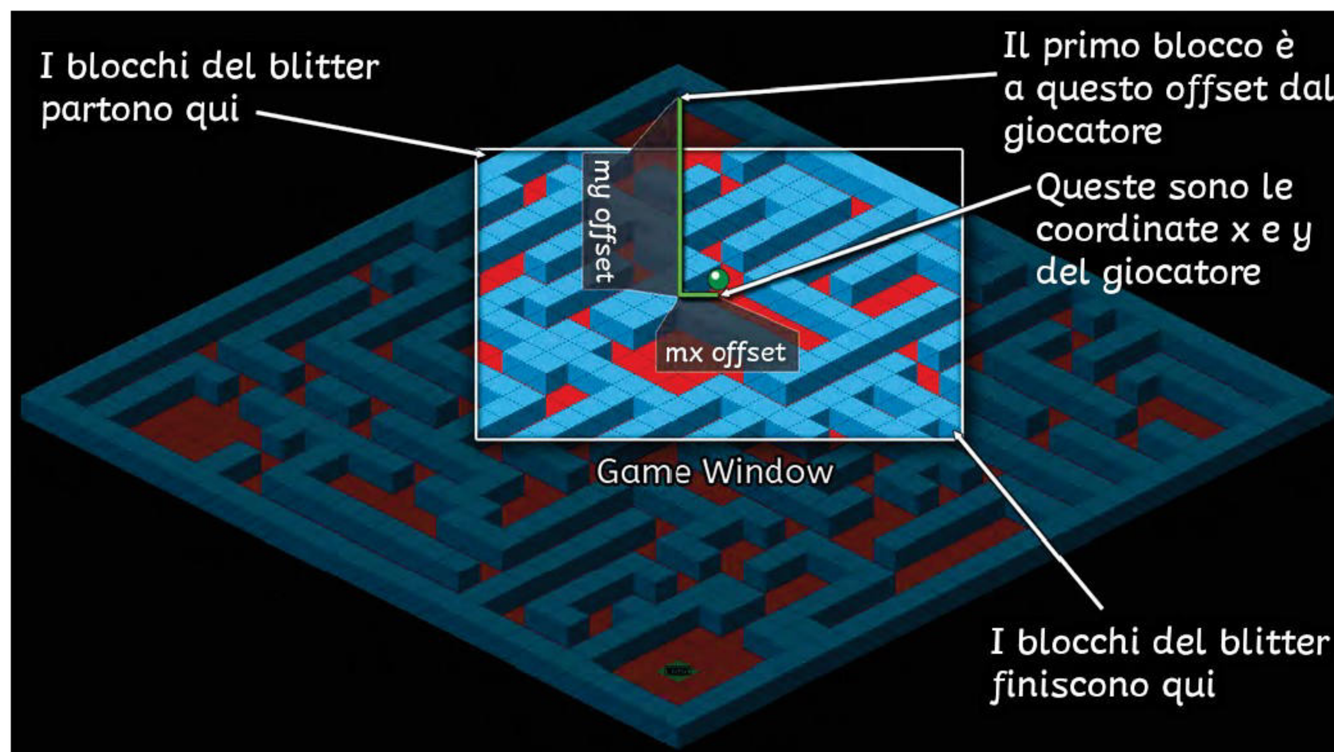


Figura 2 le coordinate da cui inizia la mappa sono calcolate come un offset dal giocatore, e i blocchi del labirinto sono disegnati solo dentro il rettangolo della finestra di gioco

bisogno. Facciamo un nuovo modulo per la gestione delle mappe, come già fatto in precedenti tutorial (vedi *The MagPi* #76, **6 &; 9d((ªúù**). Chiamiamo il nuovo modulo **map3d.py**. Python fornisce un modulo per importare file JSON, quindi possiamo scrivere `import json` all'inizio del file *6 &9ô) h9B* per caricare il modulo. Bene, è necessario utilizzare anche il modulo os per gestire il percorso dei file, quindi importa anche questo. Poi dobbiamo solo scrivere la funzione per caricare la nostra mappa.

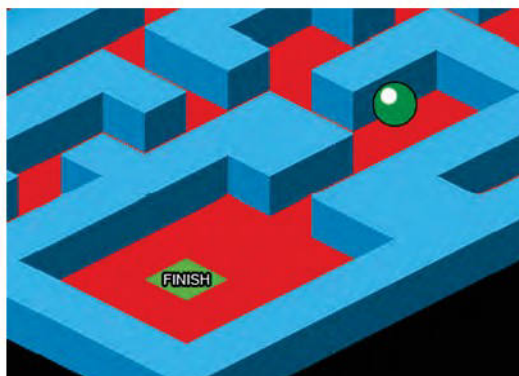
Top Tip

Vedere i JSON

Puoi vedere i file JSON con qualunque editor di testo, ma uno di programmazione è probabilmente meglio – prova Geany.

07 Ottenere ciò che serve

faremo una funzione chiamata `loadmap()` e useremo un parametro chiamato `mp` per passare il percorso del file alla funzione. Dai un'occhiata alla *+q >; *òh9B* per vedere come scriverla. Puoi vedere che carica i dati della mappa nella variabile `d` usando la funzione `json.load()`. Quindi possiamo copiare la larghezza e l'altezza



Abbiamo aggiunto una nuova tessera per finire il gioco. Quando il giocatore si sposta su questo blocco, il labirinto è risolto

in un dizionario chiamato `mapdata`. Questo dizionario conterrà tutti i dati di cui abbiamo bisogno nel lasso di tempo per arrivare alla fine della funzione. Avendo fatto una copia temporanea dei dati dei blocchi `(rawdata)`, possiamo scorrerli ed inserirli nel formato che vogliamo in `mapdata`.

08 Connessione al tileset

Una parte delle informazioni che ci servono è dove trovare quale immagine utilizzare per ciascun blocco. Queste sono comprese in un valore chiamato 'tilesets'. In questo caso assumeremo che abbiamo un solo tileset definito, in modo da leggerlo e trovare le immagini dei blocchi. Tranne che c'è un piccolo problema. I dati della mappa si riferiscono al file tileset come a un file .tsx. Quello che dobbiamo fare è tornare su Tiled ed esportare il tileset come file JSON. Quindi quando lo importiamo dobbiamo cambiare l'estensione .tsx in .json.

09 Una notte sulle tessere

Una volta caricato i dati del tileset come file JSON, possiamo scorrere le tessere e ottenere i nomi delle immagini dei blocchi. Noterai che aggiungendo uno al valore id, questo corrisponde al valore esportato da Tiled. Quando abbiamo tutti i dati nel dizionario `mapdata`, possiamo tornare al programma principale scrivendo `return mapdata`. Una volta ritornati al programma principale,

dovremo aggiungere un import per il modulo map3d all'inizio del codice e quindi, prima della funzione `draw()` possiamo scrivere `mapData = map3d.loadmap("maps/map1.json")` anziché la nostra lista di uno e zero.

10 Pensare in grande

Nella prima parte di questo tutorial, il nostro labirinto era di 12 x 12 blocchi, che si adattava bene all'area di gioco. Ora abbiamo un labirinto da 30 x 30 che, se lo disegnamo tutto, esce dai lati della finestra di gioco. Dobbiamo essere in grado di far scorrere la mappa sullo schermo mentre il giocatore si muove nel labirinto. La via da usare per ottenerlo è mantenere la palla rimbalzante al centro dello schermo e, quando si muove il giocatore, far scorrere la mappa. Quindi, in effetti, quello che stiamo dicendo è che disegneremo la mappa in relazione al giocatore piuttosto che in relazione alla finestra di gioco.

11 È tutto relativo

Per disegnare la mappa, useremo gli stessi cicli di base (x e y) di prima, ma facendo partire il disegno dalle coordinate x e y del giocatore sullo schermo. Con questa posizione della schermata iniziale, facciamo un ciclo in un intervallo che è uno dei due lati del giocatore in entrambe le direzioni. Vedi la *J ꝗ >; &ó* per una spiegazione visuale di ciò che stiamo facendo in questi cicli. In termini semplici, quello che stiamo dicendo è: 'Inizia a disegnare la mappa dalle coordinate che faranno apparire il giocatore al centro della finestra. Quindi disegna solo i blocchi che sono visibili nella finestra.' Vedi la *+ꝗ >; *ôlɧB* per capire come è cambiata la funzione `drawMap()` per fare questo.

12 Funzioni extra

Vedrai in *+ꝗ >; *ôlɧB* che abbiamo un paio di nuove funzioni che non abbiamo ancora definito. La prima si chiama `onMap()`, che passa le coordinate x e y. Questi sono blocchi che testiamo per essere sicuri di rimanere nella mappa, altrimenti otteniamo un errore. Se x o y sono inferiori a 0 o maggiori della larghezza (o altezza) della mappa, possiamo ignorarlo. L'altra funzione è `findData()`. Questa trova i dati associati ad una tessera di un determinato id. Vedi la *+ꝗ >; *őlɧB* (sul retro) per vedere come sono scritte queste funzioni.

figure3.py

```python
def drawMap():
    psx = OFFSETX
    psy = OFFSETY-32
    mx = psx - player["sx"]
    my = psy - player["sy"]+32

    for x in range(player["x"]-12, player["x"]+16):
        for y in range(player["y"]-12, player["y"]+16):
            if onMap(x,y):
                b = mapData["data"][y][x]
                td = findData(mapData["tiles"], "id", b)
                block = td["image"]
                bheight =  td["imageheight"]-34
                bx = (x*32)-(y*32) + mx
                by = (y*16)+(x*16) + my
                if -32 <= bx < 800 and 100 <= by < 620:
                    screen.blit(block, (bx, by - bheight))
                if x == player["x"] and y == player["y"]:
                    screen.blit("ball"+str(player["frame"]),
(psx, psy))
```

La funzione drawMap() aggiornata

map3d.py

Linguaggio: **Python**

```python
# Modulo 3dmap per AmazeBalls
import json
import os

def loadmap(mp):
    with open(mp) as json_data:
        d = json.load(json_data)
    mapdata = {"width":d["width"], "height":d["height"]}
    rawdata = d["layers"][0]["data"]
    mapdata["data"] = []
    for x in range(0, mapdata["width"]):
        st = x*mapdata["width"]
        mapdata["data"].append(rawdata[st:st+mapdata["height"]])

    tileset = "maps/" + d["tilesets"][0]["source"].replace(
".tsx",".json")
    with open(tileset) as json_data:
        t = json.load(json_data)

    mapdata["tiles"] = t["tiles"]
    for tile in range(0,len(mapdata["tiles"])):
        path = mapdata["tiles"][tile]["image"]
        mapdata["tiles"][tile]["image"] = os.path.basename(path)
        mapdata["tiles"][tile]["id"] = mapdata["tiles"][tile]["id"]+1
    return mapdata
```

figure4.py

```
def onMap(x,y):
    if 0 <= x < mapData["width"] and 0 <= y < mapData["height"]:
        return True
    return False

def findData(lst, key, value):
    for i, dic in enumerate(lst):
        if dic[key] == value:
            return dic
    return -1
```

Le funzioni per testare se le coordinate sono all'interno dell'area della mappa – onMap() – e findData(), che trova i dati delle tessere per il disegno della mappa -

figure5.py

```
def doMove(p, x, y):
    if onMap(p["x"]+x, p["y"]+y):
        mt =
mapData["data"][p["y"]+y][p["x"]+x]
        if mt == 1:
            p.update({"queueX":x,
"queueY":y, "moveDone":False})
            if mt == 3:
                mazeSolved = True
```

La funzione doMove() aggiornata

13 Mascherare i bordi

Se disegnamo ora la nostra mappa, abbiamo perso la bella forma di diamante. E se spostiamo il giocatore verso il basso sulla mappa, otteniamo un bordo frastagliato in alto e blocchi che appaiono e scompaiono a seconda del risultato della funzione `drawMap()`. Si può ovviare a questo sovrapponendo una cornice che oscura i bordi dell'area di disegno. Lo facciamo avendo un'immagine che copre l'intera finestra ma con un'area trasparente in corrispondenza della mappa da mostrare. Mixiamo questa cornice dopo aver chiamato `drawMap()` nella funzione `draw()`.

14 Non posso muovermi!

Ora che abbiamo disegnalo la mappa, se aggiungi il codice seguendo quello mostrato nella prima parte, non riuscirai più a muovere la palla rimbalzante. Questo perché i dati che stai leggendo sono in un formato leggermente differente e hanno i blocchi pavimento id 1 e i muri id 2, quindi al momento la funzione `doMove()` crede di essere circondata da muri (id 1 nella puntata precedente). Abbiamo bisogno di modificare la funzione `doMove()` per adattare il nuovo formato di dati. Dai un'occhiata a **figure5.py** per vedere cosa dobbiamo scrivere.

Top Tip

Ordine di disegno

Ricorda che nella funzione `draw()`, le cose sono disegnate nell'ordine di chiamata, quindi disegna sempre le cose che vuoi sopra, per ultime.

15 Trovare l'uscita

Ora che abbiamo riordinato il display e la palla si muove di nuovo, dobbiamo cambiare alcuni valori di default con i quali iniziare. Nella scorsa parte il giocatore iniziava con x = 0 e y = 3. Dobbiamo cambiare questi valori a 3 e 3 nei dati del giocatore nella parte superiore del codice, per adattarlo a questa mappa. Bisogna anche cambiare la costante `OFFSETY` a 300 per spostare la mappa un po' oltre la parte bassa dello schermo. Ora siamo in grado di spostare la palla rimbalzante nel labirinto verso la parte bassa dello schermo dove c'è la tessera finale.

16 Oltre il traguardo

Se proviamo ad andare sul blocco finale, non ne saremo in grado, perché la funzione `doMove()` può riconoscere solo blocchi con id 1. Quindi dobbiamo aggiungere un'altra condizione. Invece di verificare solo il valore 1 della variabile `mt`, aggiungiamo la linea se `mt == 1` o `mt == 3`: (perché l'id della tessera di arrivo è 3). Possiamo quindi aggiungere una variabile da impostare se il giocatore si sposta sulla tessera finale, aggiungendo la condizione: `if mt == 3: mazeSolved = True`. Dobbiamo dichiarare `mazeSolved` globale dentro `doMove()` e impostare il suo valore iniziale a `False` all'inizio del nostro programma.

17 Il tempo sta finendo

Aggiungiamo un timer al gioco. Quando il giocatore raggiunge il traguardo (`mazeSolved is True`), possiamo fermare il timer e mostrare un messaggio. Quindi, prima creiamo una variabile timer all'inizio del programma con `timer = 0` e poi, proprio alla fine del codice, poco prima di `pgzrun.go()`, possiamo usare la funzione orologio di Pygame Zero chiamata `schedule_interval()`. Se scriviamo `clock.schedule_interval(timerTick, 1.0)`, la funzione `timerTick()` verrà chiamata una volta al secondo.

18 L'orologio suona

Quel che ora dobbiamo fare è definire la funzione `timerTick()`. Ci serve controllare se la variabile `mazeSolved` è `False` e, se lo è, aggiungere 1 alla variabile `timer`. Poi, possiamo aggiungere la riga `screen.draw.text` alla funzione `draw()` per mostrare il valore del timer, e se `mazeSolved` è `True`, possiamo aggiungere altro testo per dire che il labirinto è stato risolto e in quanto tempo. Vedi il listato completo del programma per vedere come scrivere il codice per questi eventi.

Nella prossima puntata, aggiungeremo alcuni nemici da combattere e, giusto per buona misura, potremo lanciare un po' di dinamite!



amazeballs2.py

SCARICA IL CODICE COMPLETO: magpi.cc/fPBrhM

› Linguaggio: **Python**

```
import pgzrun
import map3d

player = {"x":3, "y":3, "frame":0, "sx":0, "sy":96,
          "moveX":0, "moveY":0, "queueX":0, "queueY":0,
          "moveDone":True, "movingNow":False, "animCounter":0}
OFFSETX = 368
OFFSETY = 300
timer = 0
mazeSolved = False

mapData = map3d.loadmap("maps/map1.json")

def draw(): # Funzione disegno Pygame Zero
    screen.fill((0, 0, 0))
    drawMap()
    screen.blit('title', (0, 0))
    screen.draw.text("TIME: "+str(timer) , topleft=(20, 80), owidth=0.5, ocolor=(255,255,0), color=(255,0,0) , fontsize=60)
    if mazeSolved:
        screen.draw.text("MAZE SOLVED in " + str(timer) + " seconds!" , center=(400, 450), owidth=0.5, ocolor=(0,0,0), color=(0,255,0) , fontsize=60)


def update(): # Funzione aggiornamento Pygame Zero
    global player, timer
    if player["moveDone"] == True:
        if keyboard.left: doMove(player, -1, 0)
        if keyboard.right: doMove(player, 1, 0)
        if keyboard.up: doMove(player, 0, -1)
        if keyboard.down: doMove(player, 0, 1)
    updateBall(player)

def timerTick():
    global timer
    if not mazeSolved:
        timer += 1

def drawMap():
    psx = OFFSETX
    psy = OFFSETY-32
    mx = psx - player["sx"]
    my = psy - player["sy"]+32

    for x in range(player["x"]-12, player["x"]+16):
        for y in range(player["y"]-12, player["y"]+16):
            if onMap(x,y):
                b = mapData["data"][y][x]
                td = findData(mapData["tiles"], "id", b)
                block = td["image"]
                bheight =  td["imageheight"]-34
                bx = (x*32)-(y*32) + mx
                by = (y*16)+(x*16) + my
                if -32 <= bx < 800 and 100 <= by < 620:
                    screen.blit(block, (bx, by - bheight))
                if x == player["x"] and y == player["y"]:
                    screen.blit("ball"+str(player["frame"]), (psx, psy))

def findData(lst, key, value):
    for i, dic in enumerate(lst):
        if dic[key] == value:
            return dic
    return -1

def onMap(x,y):
    if 0 <= x < mapData["width"] and 0 <= y < mapData["height"]:
        return True
    return False

def doMove(p, x, y):
    global mazeSolved
    if onMap(p["x"]+x, p["y"]+y):
        mt = mapData["data"][p["y"]+y][p["x"]+x]
        if mt == 1 or mt == 3:
            p.update({"queueX":x, "queueY":y, "moveDone":False})
            if mt == 3:
                mazeSolved = True

def updateBall(p):
    if p["movingNow"]:
        if p["moveX"] == -1: moveP(p,-1,-0.5)
        if p["moveX"] == 1: moveP(p,1,0.5)
        if p["moveY"] == -1: moveP(p,1,-0.5)
        if p["moveY"] == 1: moveP(p,-1,0.5)
    p["animCounter"] += 1
    if p["animCounter"] == 4:
        p["animCounter"] = 0
        p["frame"] += 1
        if p["frame"] > 7:
            p["frame"] = 0
        if p["frame"] == 4:
            if p["moveDone"] == False:
                if p["queueX"] != 0 or p["queueY"] !=0:
                    p.update({"moveX":p["queueX"], "moveY":p["queueY"], "queueX":0, "queueY":0, "movingNow": True})
            else:
                p.update({"moveDone":True, "moveX":0, "moveY":0, "movingNow":False})

        if p["frame"] == 7 and p["moveDone"] == False and p["movingNow"] == True:
            p["x"] += p["moveX"]
            p["y"] += p["moveY"]
            p["moveDone"] = True

def moveP(p,x,y):
    p["sx"] += x
    p["sy"] += y

clock.schedule_interval(timerTick, 1.0)
pgzrun.go()
```

Sviluppare una app Android

Puoi creare un'app solo con il Raspberry Pi e installarla sul tuo telefono Android? Certo che puoi, ed ecco come...

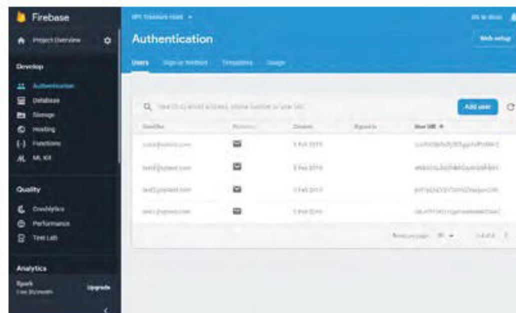


MAKER
Mark Vanstone

Autore di software educativo degli anni '90, autore della serie ArcVenture, scomparso nella landa desolata del software aziendale. Salvato dal Raspberry Pi!

magpi.cc/YiZnxL

Raspberry Pi è generalmente considerato come una grande piattaforma per imparare come programmare e controllare progetti elettronici, ma puoi creare programmi da usare su altre Piattaforme, come PC o anche telefoni cellulari? Bene, considerando il costo d'acquisto di un Raspberry Pi, potresti pensare che sia un po' limitato per lo sviluppo reale, ma questo articolo ti spiegherà come usare il Pi per creare una app mobile reale, data-driven e utile, che puoi installare su un telefono e utilizzare per comunicare in tempo reale con il tuo Raspberry Pi.



Nella console Firebase, sotto Authentication, puoi impostare gli utenti per la tua app

Esistono diversi metodi per creare una app mobile. Generalmente, il modo per produrre un'app è scriverla usando un sistema di sviluppo che compila il tuo codice in un pacchetto per una specifica piattaforma. Queste sono chiamate app "native" e dovresti compilare diverse versioni per telefoni Android, iPhone o telefoni Windows. Ci sono altre app che hanno uno wrapper app nativo, ma poi visualizzano pagine HTML5 all'interno del wrapper o può essere che lavorino anche solo all'interno di un browser web. Queste sono chiamate app "web".

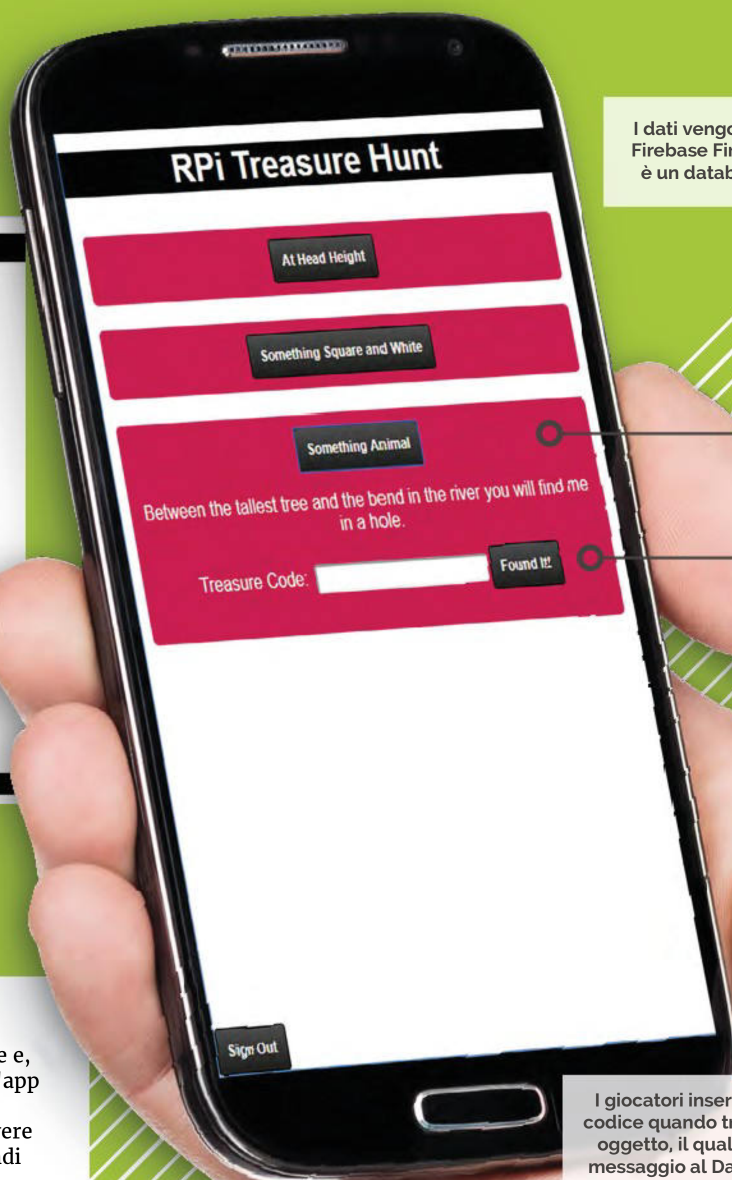
Il problema con le app Web è che si basano sul fatto di avere una connessione a internet per visualizzare le pagine HTML e qualsiasi trattamento andrà generalmente a richiedere chiamate a script e database lato server come fanno le normali pagine web. Ora, c'è una soluzione a questo problema e si chiama 'PWA'. Sta per 'progressive web app' e ha elementi del modo in cui funziona un'applicazione nativa, ma può essere programmata usando la codifica della pagina web.

Un PWA può mantenere una copia delle schermate dell'app e anche una copia locale dei dati del database in modo che quando non c'è la connessione a internet, continui a funzionare. Può anche essere installato su un telefono cellulare,

Puoi assegnare i dettagli di accesso per i tuoi utenti nella console di Firebase



Quando l'app si avvia, controlla se ha avviato il framework Firebase



I dati vengono letti dal Firebase Firestore, che è un database NoSQL

I giocatori inseriscono un codice quando trovano un oggetto, il quale invia un messaggio al Database in tempo reale di Firebase

con una propria icona sulla schermata iniziale e, una volta avviata, sembri e funzioni come un'app nativa.

Per scrivere una PWA sul Raspberry Pi, e avere un sistema per acquisire i dati dall'app e quindi fare in modo che l'elettronica esterna faccia qualcosa in risposta, avremo l'aiuto di un framework di Google, chiamato Firebase.

> "Faremo un'app caccia al tesoro per telefoni mobili"

Quindi ecco il piano: usando Raspbian, installeremo i moduli necessari sul Raspberry Pi e imbastiremo un nuovo progetto Firebase. Faremo un'app di caccia al tesoro per telefoni mobili che comunicherà con un programma in Python sul Raspberry Pi. Quel programma userà i pin GPIO per illuminare delle luci colorate per ogni squadra che cerca o completa la caccia al tesoro. Tutto questo può essere sviluppato e implementato da Raspberry Pi!

CHI STA SOSTENENDO PWA?

Questo è un esempio molto semplice e gli obbiettivi di ciò che può essere fatto con le progressive web app, sono molto più grandi. Per molti progetti di app mobili, questo framework sarà sufficiente e probabilmente dobbiamo chiederci se è necessario inviare le app all'App Store o al Play Store (e il costo sostenuto in questo caso) quando con un semplice link può essere inviata a un utente che può installare un'app funzionale che sembra e funziona come un'app nativa. C'è una vasta documentazione sul sito Firebase che copre tutti gli elementi del framework. Google si impegna a supportare la tecnologia PWA e anche Microsoft ha detto che la sosterrà. Anche se Apple è meno loquace sul suo supporto, un collegamento PWA installa un'app su iPhone e il PWA agisce come una app nativa per l'utente. Per vedere un elenco di quello che è disponibile tramite una PWA, guarda il grafico su **whatwebcando.today**. Tranne qualche funzione dipendente dall'hardware specifico di un dispositivo mobile, la lista offre una vasta gamma di funzioni che senza dubbio si espanderanno man mano che vengono create e utilizzate più PWA.

Creare l'app

Una guida passo-passo per realizzare la tua app mobile

Cosa Serve

- Raspbian (ultima versione)
- Node.js **magpi.cc/ogOWTs**
- firebase-tools **firebase.google.com**
- Modulo Python Pyrebase **magpi.cc/OPQUtg**
- Telefono Android
- LED, resistenze, cavallotti e breadboard

01 La giusta versione di Node.js

La prima cosa da fare è installare Node.js. Sfortunatamente, la versione attualmente disponibile tramite il comando apt-get non è sufficientemente aggiornata, quindi ci servirà fare un'installazione manuale. Vai su **magpi.cc/ogOWTs** e scarica il pacchetto per ARM. Per scoprire di quale versione necessiti, digita `uname -m` in una finestra del Terminale. La maggior parte dei modelli di Pi avranno bisogno del pacchetto v7. Dovrai decomprimere il file in una directory appropriata (la directory home va benone) e poi, nella finestra del Terminale, vai alla directory decompressa con, per esempio, `cd node-v10.15.1-linux-armv71`. Adesso scrivi `sudo cp -R * /usr/local/` per copiare i file.

02 Recupera Firebase tools

Firebase usa Node.js per il suo set di strumenti e una volta che hai i file Node nella giusta posizione, puoi controllare che siano pronti per l'uso digitando `node -v`, che dovrebbe rispondere con la versione che hai appena installato (10.15.1 nel nostro caso). Puoi anche controllare il Node Package Manager digitando `npm -v`. Ci servirà npm per installare gli strumenti di Firebase. Lo facciamo digitando `sudo npm install -g firebase-tools` nella finestra del Terminale. L'installazione impiegherà un po' di tempo; quando è terminata, esegui nuovamente lo stesso comando, per aggiornare all'ultimo pacchetto.

03 Ottieni un account Firebase

Google, molto gentilmente, fornisce l'accesso di base al suo servizio Firebase, gratis. Tutto quel che devi fare è registrare un account su **firebase.google.com** e sarai in grado di fare tutte le cose indicate in questo tutorial. Puoi farlo accedendo con il tuo account Google esistente, se ne hai uno. Google offre anche diversi piani a pagamento se avrai bisogno di espandere le tue esigenze. Una volta ottenuto l'account, sarai in grado di accedere alla console Firebase all'indirizzo **console.firebase.google.com**. Sulla pagina principale della console, vedrai un'opzione per aggiungere un nuovo progetto.

04 Il nuovo progetto

Seleziona il pulsante "Aggiungi progetto" e potrai configurare il tuo progetto Firebase. Dovrai digitare un nome del progetto e accettare i termini e le condizioni, quindi seleziona "Crea progetto". Quando il progetto verrà creato, sarai indirizzato alla Firebase console dove vedrai una serie di strumenti sul lato sinistro. Useremo gli strumenti del gruppo "Sviluppo" che dovrebbe essere presente qui. In primo luogo, diamo un'occhiata a Autenticathion. Dovrai impostare un "Metodo di accesso", per cui è possibile abilitare "Email/Password". Poi, nella scheda "Utenti", puoi aggiungere un utente. In questa fase, vai anche nella sezione Database e crea un database Firestore.

05 Archiviazione

Firebase offre tre categorie diverse di archiviazione dei dati. la prima è l'archiviazione del database da cui possiamo usare Firestore o Realtime Database (lo approfondiremo più tardi). La seconda è etichettata come 'Storage' nel menu

manifest.json

> Linguaggio: **JSON**

```
{
    "short_name": "RPiTreasure",
    "name": "RPi Treasure Hunt",
    "icons": [
      {
        "src": "/images/rpit192.png",
        "type": "image/png",
        "sizes": "192x192"
      },
      {
        "src": "/images/rpit512.png",
        "type": "image/png",
        "sizes": "512x512"
      }
    ],
    "start_url": "/",
    "background_color": "#fff",
    "display": "standalone",
    "scope": "/",
    "theme_color": "#fff"
  }
```




treasure.py

Linguaggio: **Python**

SCARICA IL CODICE COMPLETO


magpi.cc/OdnyDP

```python
import pyrebase
import time
from gpiozero import LED

config = {
    "apiKey": "Your apiKey goes here",
    "authDomain": "Your hosting domain goes here",
    "databaseURL": "Your hosting URL goes here",
    "projectId": "Your project id",
    "storageBucket": "Your storage domain",
    "messagingSenderId": "Your sender id"
}

firebase = pyrebase.initialize_app(config)
numberOfTreasure = 3
led = {}
teams = {}
teams[0] = {"email":"test1@rpitest.com", "led":17,"treasure":[]}
teams[1] = {"email":"test2@rpitest.com", "led":18,"treasure":[]}
teams[2] = {"email":"test3@rpitest.com", "led":22,"treasure":[]}
teams[3] = {"email":"test4@rpitest.com", "led":23,"treasure":[]}

for f in range(len(teams)):
    led[f] = LED(teams[f]["led"])
    led[f].off()

db = firebase.database()

def processMessage(d):
    if(d != None):
        for v in d.values():
            updateTeam(v["email"], v["item"])

def ledFlash(t):
    for f in range(5):
        led[t].on()
        time.sleep(.2)
        led[t].off()
        time.sleep(.2)

def ledOn(t):
    led[t].on()

def updateTeam(t,i):
    for td in teams:
        if teams[td]["email"] == t:
            if i not in teams[td]["treasure"]:
                teams[td]["treasure"].append(i)
                if len(teams[td]["treasure"]) >= numberOfTreasure:
                    print(t+" complete!")
                    ledOn(td)
                else:
                    ledFlash(td)

def streamHandler(message):
    if message["event"] == "put" or message["event"] == "patch":
        processMessage(message["data"])

myStream = db.child("msg").stream(streamHandler)

while 1:
    time.sleep(.1)
```

che è un'area per memorizzare i file generati da un'app. la terza, che vedremo ora, è 'Hosting'. Se andiamo nella sezione Hosting, vedremo le istruzioni sull'installazione di firebase-tools, cosa che abbiamo fatto in precedenza. Poi ci sono le istruzioni su come impostare il progetto sul tuo Raspberry Pi - quindi seguiamole.

06 Progetti locali

Per prima cosa, nella nostra finestra del Terminale, creiamo una directory per la nostra app con `mkdir nomeapp`. Poi `cd nomeapp` per entrare in quella directory. Dalle istruzioni di Firebase, digitare `firebase login`. Apparirà una finestra del browser e ti verrà chiesto di accedere al tuo account Google/Firebase. Una volta fatto, torna alla finestra del Terminale e digita `firebase init`. Ti verrà chiesto quali funzionalità ti piacerebbe usare, che potrebbero essere tutte, per il momento. Ti chiederà di selezionare il tuo progetto, poi ti farà una serie di domande sul tuo progetto. Basta selezionare l'opzione predefinita (premere **INVIO**) per tutte le domande.

07 Setup fatto

Ora Firebase ha impostato un progetto predefinito per noi, che ha tutto il necessario per costruire la nostra app. Torna alla console di Firebase e seleziona "Finito" (faremo il deploy un po' più tardi). All'interno della nostra cartella app troveremo un'altra cartella chiamata **public**. Dentro di essa abbiamo il nostro file **index.html**,

sw.js

Linguaggio: **JavaScript**

```

var cacheName = 'rpitreasure';
var filesToCache = [
  '/',
  '/index.html',
  '/images/logo.png',
];

self.addEventListener('install', function(e) {
  console.log('[ServiceWorker] Install');
  e.waitUntil(
    caches.open(cacheName).then(function(cache) {
      console.log('[ServiceWorker] Caching app shell');
      return cache.addAll(filesToCache);
    })
  );
});

self.addEventListener('activate',  event => {
  event.waitUntil(self.clients.claim());
});

self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request, {ignoreSearch:true}).then(response => {
      return response || fetch(event.request);
    })
  );
});
```

ed è dove la nostra app sarà realizzata. Diamo un'occhiata a questo file: apri il tuo editor di programmazione preferito (potresti provare Geany se sei indeciso) e carica il file **index.html**. Ora vai alla console di Firebase, seleziona l'ingranaggio vicino a "Project overview" e seleziona "Impostazioni progetto".

08 Aggiungere un po' di Firebase

Nella pagina Impostazioni progetto, verso il basso vedrai un pannello che dice che non ci sono app nel tuo progetto. Seleziona l'icona rotonda del web (</>) e otterrai un pop-up con del codice JavaScript. Copia quel codice e torna a modificare il tuo file **index.html**. Ora sostituisci il codice che inizia con `<!-- update the version number` e termina con `/init.js> </script>` con il codice che hai appena copiato dalla console Firebase. Questo pezzo di il codice collegherà la nostra app Web con i servizi Firebase. Ora possiamo fare il deploy della nostra app di test, tornando alla nostra finestra del Terminale e digitando `firebase deploy`. Dopo aver caricato i file necessari, l'app sarà pronta per essere testata su un browser.

09 Testing testing

Possiamo testare sul browser Chromium di Raspberry Pi o su un dispositivo mobile, ma assicurati di utilizzare Chrome o Chromium. Altri browser supportano i PWA, ma per ora manteniamolo semplice. Dopo che è stato fatto il deploy del progetto, riceveremo l'indirizzo dell'URL di hosting nella nostra finestra del Terminale. Punta un browser a questo indirizzo e dovresti vedere conferma che l'app funziona e è connessa. Puoi anche testarla localmente se hai installato l'estensione Chrome Web Server. Se non vedi un messaggio che dice "Firebase SDK loaded with auth, database, messaging, storage", allora torna indietro ai passaggi precedenti.

10 Creiamo l'app

La prima cosa da programmare sarà l'app per il telefono Dai un'occhiata al listato di **index.html**. Quel che fa è presentare una

TUTTO QUESTO VALE PER LE APP REALI?

Naturalmente, ci sarà qualcuno che sostiene che se non si tratta di un'app nativa e quindi non è un'app reale e le restrizioni di App Store e Play Store attuali fanno sì che sia difficile adattare una PWA per la distribuzione con questi canali. Il fatto è che, nell'industria, il costo di qualsiasi progetto è fondamentale perché prenda forma e, mentre lo sviluppo di app native è un'attività molto costosa, le PWA possono essere programmate in una frazione del tempo e non hanno restrizioni sull'App Store o Play Store. Questo non vuol dire che le PWA non possano essere pubblicate su quei portali, è solo che devono essere wrappate in un framework come Cordova per trasformarle in app native.

Per la comunità dei maker/coder, la possibilità di pubblicare app senza queste restrizioni è senza dubbio un vantaggio e possibilmente una tecnologia che lo farà, soppianterà i negozi dipendenti dalla piattaforma. Se non altro, la tecnologia PWA ci dà una grande opportunità per creare app utili con solo un Raspberry Pi.

index.html

Linguaggio: **HTML**

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="manifest" href="/manifest.json">
    <title>Welcome to The RPi Treasure Hunt</title>
<script src="https://www.gstatic.com/firebasejs/5.8.1/firebase.js"></script>
<script>
  // Inizializza Firebase
  var config = {
    apiKey: "La tua apiKey va qui",
    authDomain: "Il tuor hosting domain va qui",
    databaseURL: "L’URL del tuo database va qui",
    projectId: "Il tuo project id",
    storageBucket: "Il tuo storage domain",
    messagingSenderId: "Il tuo sender id"
  };
  firebase.initializeApp(config);
</script>
<script>
  if(‘serviceWorker’ in navigator) {
    navigator.serviceWorker.register(‘/sw.js’)
      .then(function() {
            console.log(‘Service Worker Registered’);
      });
  }
</script>
    <style media="screen">
      body { background: #fff; color: #000; font-family: Helvetica, Arial, sans-serif; margin: 0; padding: 0; text-align: center; background: url(images/logo.png) no-repeat 50% 300px fixed; background-size: 50% }
      #title{ background: #000;color:#fff}
      #signin{ margin:10px; padding:10px;}
      #signout{ position: fixed; bottom: 0px;text-align: center}
      .clueholder{ background:#ca0d4c; color:#fff; border:10px solid #fff; margin:0px; padding:10px;-webkit-border-radius: 15px; -moz-border-radius: 15px;border-radius: 15px;}
      .theClue{display:none}
      #loginform{background: #ca0d4c;color:#fff;padding:10px;text-align: right;width:300px;margin: auto;-webkit-border-radius: 3px; -moz-border-radius: 3px;border-radius: 3px;}
      #load {
        display: block; text-align: center; background: #000; text-decoration: none; color: white;
        position: fixed; bottom: 0;padding-top: 5px;padding-bottom: 5px;
        width: 100%; height:18px;
      }
      input{ -webkit-border-radius: 3px; -moz-border-radius: 3px;border-radius: 3px;margin:3px;padding:2px}
      button {
        border:1px solid #000; -webkit-border-radius: 3px; -moz-border-radius: 3px;border-radius: 3px;font-size:12px;font-family:arial, helvetica, sans-serif; padding: 10px 10px 10px 10px; text-decoration:none; display:inline-block;font-weight:bold; color: #fff;
        background-image: -webkit-gradient(linear, left top, left bottom, from(rgb(77, 77, 77)), to(rgb(29, 29, 27)));;
      }
    </style>
  </head>
  <body>
    <div id="title">
      <h1>RPi Treasure Hunt</h1>
    </div>
    <div id="content">
      Loading RPi Treasure Hunt App.
    </div>
    <p id="load">Connecting ...</p>
    <script>
      clues = null;
      email = "";
      document.addEventListener(‘DOMContentLoaded’, function() {
        try {
          let app = firebase.app();
          let features = [‘auth’, ‘database’, ‘messaging’, ‘storage’].filter(feature => typeof app[feature] === ‘function’);
          document.getElementById(‘load’).innerHTML = `Connected to Treasure Hunt.`;
          firebase.firestore().enablePersistence();
        } catch (e) {
          console.error(e);
          document.getElementById(‘load’).innerHTML = ‘Error connecting to the Treasure Hunt.’;
        }
      });
      firebase.auth().onAuthStateChanged(function(user) {
        window.user = user; // l’utente è indefinito se nessun utente è loggato
        console.info("user changed - is now "+user);
```

```
        if (user == null){
          setLoginPage();
        }else{
          document.getElementById(‘load’).style = "display:none"
          getClueData();
        }
      });
      function signinUser(){
        email = document.getElementById("email").value
        password = document.getElementById("pass").value
        firebase.auth().signInWithEmailAndPassword(email, password)
        .catch(function(err) {
          console.error(err);
        });
      }

      function getClueData(){
        var db = firebase.firestore();
        db.collection("Clues").get().then(function(querySnapshot) {
          setCluesPage(querySnapshot);
        });
      }

      function signoutUser(){
        firebase.auth().signOut()
        .catch(function (err) {
          console.error(err);
        });
      }

      function openThisClue(o){
        cluelist = document.getElementsByClassName(‘theClue’)
        for(c=0;c<cluelist.length;c++){
            cluelist[c].style = "display:none";
        }
        theclue = document.getElementById(o);
        theclue.style = "display:block";
      }

      function foundItem(i){
        code = document.getElementById("clueCode_"+i).value;
        clues.forEach(function(doc){
          if(doc.id == i && code == doc.data().code){
            console.log("we have a winner");
            var newMsgKey = firebase.database().ref().child(‘msg’).push().key;
            var postData = {
              email: email,
              item: i
            };
            var updates = {};
            updates[‘/msg/’ + newMsgKey] = postData;
            firebase.database().ref().update(updates);
            document.getElementById("clue_"+i).style = "display:none"
          }
        });
      }

      function setLoginPage(){
        document.getElementById(‘content’).innerHTML = "<div id=’loginform’>Email: <input id=’email’ type=’text’><br>Password: <input id=’pass’ type=’password’><br><button onclick=’signinUser();’>Sign In</button></div>";
      }

      function getClueDisplay(cluelist){
        out = "";
        clues = cluelist;
        clues.forEach(function(doc){
          out += `<div class="clueholder" id="clue_`+doc.id+`"><button onclick="openThisClue(‘clueDetail_`+doc.id+`’)">`+doc.data().name+`</button><div id="clueDetail_`+doc.id+`" class="theClue"><p>`+doc.data().clue+`</p><p>Treasure Code: <input id="clueCode_`+doc.id+`" type="text"><button onclick="foundItem(‘`+doc.id+`’)">Found It!</button></p></div></div>`;
        });
        return out;
      }

      function setCluesPage(clues){
        out = getClueDisplay(clues);
        out += "<div id=’signout’><button onclick=’signoutUser()’>Sign Out</button></div>";
        document.getElementById(‘content’).innerHTML = out;
      }
    </script>

  </body>
</html>
```

pagina di accesso per consentire agli utenti (definiti nel nostro Firebase Autentication) di accedere e quindi ottenere una lista di indizi dal database Firebase Firestore e visualizzarli. Quando un utente trova un oggetto del tesoro, ci sarà un numero di codice (lo imposteremo nel nostro Firestore) che, se inserito correttamente, rimuoverà quell'indizio dalla lista e invierà un messaggio al Realtime Database di Firebase. Esistono due tipi di database che Firestore supporta: il Firestore che contiene i nostri dati della caccia al tesoro, e il Realtime Database.

11 Realtime Database

Il Realtime Database attiverà un evento se un altro programma è in ascolto quando i dati cambiano. Ecco come trasferiremo i dati al nostro Raspberry Pi. Dobbiamo fare una modifica alle regole di sicurezza per renderlo semplice. Nota che questa non è una buona idea per scopi 'seri', ma possiamo cambiare la sicurezza in modo da poter leggere e scrivere nel database senza che sia necessaria l'autenticazione. Lo facciamo andando al Realtime Database dalla console Firebase, selezionando la scheda 'Rules' e cambiando il valore '.rread' in **true** – facciamo lo stesso con il valore '.wwrite'. Normalmente vorremmo avere un po' più di sicurezza a protezione di un database, ma per questo esempio, semplifichiamo le cose.

12 corretto comportamento dell'app

Ci sono un paio di altre cose che dobbiamo aggiungere per rendere la nostra app installabile su un telefono cellulare. La prima è il file **sw.js**. È Un file di servizio e ci consente di memorizzare i file nella cache in modo che non serva una connessione internet per caricarli. Vedrai lo script per registrare questo file in **index.html**. L'altra è un file manifest che consente alla app di essere installata e produrrà un messaggio che chiede all'utente se vuole installarla. Il file manifest è linkato nella parte superiore di **index.html**.

13 Python

Abbiamo bisogno di raccogliere i dati che vengono crearti dalla nostra app – potremmo farlo con una semplice pagina web sul nostro Raspberry Pi, ma sarebbe noioso. Perché non avere un indicatore di punteggio con luci lampeggianti e altre cose del genere? Per farlo, ci servirà scrivere un programma in Python. Dai uno sguardo a **treasure.py** e guarda come lo facciamo... Ma aspetta: dobbiamo avere accesso a un modulo chiamato pyrebase, che è un wrapper per le funzioni Firebase che le rende davvero facili. Per installarlo, basta digitare **pip3 install pyrebase** in una finestra del Terminale.

14 Cablaggio

Ad essere onesti, potresti avere qualsiasi tipo di grande schema per un tabellone dei punteggi, ma possiamo collegare un LED per ogni squadra o giocatore della nostra caccia al tesoro. Vedi lo schema di cablaggio che siamo usando (**Figura 1**), dove il LED delle squadre lampeggerà mentre cercano il tesoro e manterremo il LED acceso quando hanno finito. Se hai molti giocatori, potresti usare strisce LED o anche una realizzazione molto più grande.

15 Questo che cosa significa?

Quello che abbiamo fatto è creare una app reale per dispositivi mobili con solo un Raspberry Pi da 35$ e un account Google gratuito. Questo è un modo di sviluppare abbastanza nuovo e, mentre veniva scritto questo articolo, il servizio Firestore è passato dalla versione beta a quella stabile. Ciò significa che le cose potrebbero cambiare abbastanza rapidamente, quindi potrebbe essere necessario fare riferimento al Sito Web di Firebase per gli aggiornamenti.

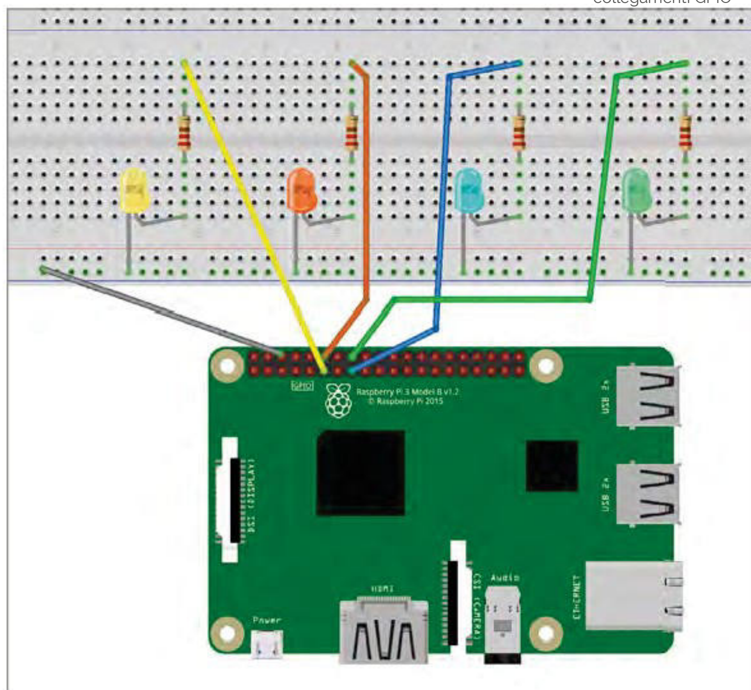


Figura 1 Come collegare quattro LED. Puoi far lampeggiare il LED quando i dati vengono ricevuti, usando i collegamenti GPIO